dLETTER

The Newsletter for dBase II and III users. Volume 1 Issue 9.

dVeloper

Menus: Their Structure and Use Part III

By Christopher White

The first two parts of this article appeared in the previous two editions of dLetter. This is the final part.

The Confirming Menu

The confirming menu is a binary choice menu. You present two mutually-exclusive choices to the user. For example,

```
 Delete this record (Y/N)?
```

or

```
 Quit to DOS (Y/N)?
```

The selection is usually a yes or no decision that confirms the user would like to continue with a choice selected from a list menu.

The confirming menu has three parts which are the same as the list menu type.

1. Display prompt and possible choices.
2. Get the user's selection.
3. Perform an action based on the selection.

The program in Listing 1 demonstrates how this works.

If you have dBase III Plus, the "Y" template of the PICTURE clause will make the code much more concise, as in Listing 2.

dLetter is published monthly by the Federal Publishing Company Pty Ltd. Printed by ESN - The Litho Centre, Waterloo 2017. All correspondence should be addressed to: The Federal Publishing Company, PO Box 21, Waterloo 2017.
Phone (02) 663 9999.

Prompting Menu

A prompting menu is a fill-in the blank type of menu. In response to a prompt the user types the choice into an entry area. For example:

```
Enter account:
                  ^--Type response here.
```

Like the confirming menu type, it has three basic parts:

1. Display the entry prompt.
2. Get user's entry.
3. Perform some action based on the user's entry.

Unlike the list and confirming menu types, the prompting menu requires more specific input. The most important issue is what to do with an invalid entry. You must perform a validation check on the user's entry and if the entry is within acceptable boundaries, perform some action. The other issue is what to do in the instance the user enters an invalid or null choice. Do you want the user to re-enter or RETURN to the calling routine? With a null entry, it is best to RETURN to the calling routine. This is a very convenient way to allow a path of escape if the user does not wish to make an entry. With an invalid entry, prompt the user again for the choice.

See Listing 3 for an example.

Design Considerations

Consistency

Along with constructing a natural flow of control, consistency is an absolute must. Thi means prompts should appear in the same place on the screen every time a prompt is displayed. It is important to retain a similarity of structure from one menu to another, deviating only if a specific task requires it.

Messages

Prompts can vary from flowery descriptions and instructions to very terse ones. Some programmers and users prefer the complete sentence approach. Here, every message is a complete English sentence with subject, predicate, and punctuation. For example:

```
Do you want to back up these
files to the A: drive (Yes or
No)?
```

or

```
We are now copying files to the
A: drive, please wait
```

The problem with this approach is the user has to read a full sentence to understand the meaning of a simple and easily understood action. After becoming familiar with the program, the user no longer needs to read a complete sentence to understand a message. A simple prompt that describes the action and the appropriate response is sufficient.

For example, the following shows the same messages, stated a little more succinctly:

```
 Back up to A: (Y/N)?
```

or

```
 Copying to A: ...
```

Several factors contribute to the brevity of messages. First is the context. Actions should be appropriate to the context in which they operate. Status messages then have a contextual reference the user understands. Users who are unfamiliar with the program or the action being performed may find this uncomfortable. You will find most major pieces of commercial software use contextual help to assist the beginning user while optimizing the operation of the program for intermediate and advanced users.

Usability

Locate status information where the eye rests naturally. Additionally, design the menu screen so the flow of operation moves the user's eye in a natural way, top to bottom or left to right. There is nothing worse than operating a menu in which the next prompt appears in an awkward location on the screen. If the last action is at the bottom of the screen, put the next set of actions near the bottom of the screen.

Trapping errors

There is some controversy whether or not it is imperative to display an error message when an invalid menu choice has been made. My personal opinion is it is not. Forcing the user to acknowledge a simple error slows down program execution and, in some ways, degrades the intelligence of the user.

There are instances where acknowledgment is necessary if there is some ambiguity involved. In some menu systems, certain choices listed on the menu may not be available depending on whether or not some other action has been performed. In this instance, informing the user of a selection that is not available is important. A beep is probably more appropriate for an incorrect key-press.

Conclusion

The essential concept I have presented in this article is that a menu-driven system can provide a meaningful user-interface for an application, when it is carefully constructed. There are three types of menus: list, prompting, and confirming, the most useful being the list menu. Common to all menu structures are three procedures: (1) displaying the list of options, (2) obtaining the user's selection, and (3) acting on the selection.

Examples of confirming menus.

```
* ---Display prompt and possible choices.
@ 23,00 SAY "Perform this action (Y/N)?"
col = COL() + 1
choice = ""
* ---Get the user's selection.
DO WHILE .NOT. choice $ "Y/N"
* ---Set the default value.
   choice = "N"
   @ 23, col GET choice PICTURE "!"
   READ
ENDDO
* ---Perform an action based on the selection.
IF choice = "Y"
   DO Yesaction
ELSE
   DO Noaction
ENDIF
```

Listing 1.

```
* ---Display prompt and get user's selection.
choice = "N"
@ 23,00 SAY "Perform this action (Y/N)?" GET choice PICTURE "Y"
READ
* ---Perform an action based on the selection.
IF choice = "Y"
   DO Yesaction
ELSE
   DO Noaction
ENDIF
```

Listing 2.

Example of action on null or invalid entry in a prompting menu.

```
filedbf = SPACE(8)
DO WHILE .T.
* ---Display prompt and get user's entry.
   @ 20,00
   @ 20,00 SAY "Enter filename:"
   GET filedbf
   READ
   DO CASE
      CASE filedbf = SPACE(8)
* ---Null entry.
         EXIT
      CASE FILE( filedbf + ".DBF" )
* ---Valid entry.
         USE &filedbf
         EXIT
   OTHERWISE
* ---Invalid entry.
      @ 20,00  WAIT TRIM( filedbf )+" not found, press <SPACE>" TO null
   ENDCASE
ENDDO
```

Listing 3.

dFunct The dBase Function of the Month

The Test ( ) Function

By Les Bell

Parsing user-input lines

In dBase II the TEST() function (the equivalent in dBase III is the TYPE() function) can be used to test the parsability of a user-input expression. Note this only tests the grammatical correctness of the expression; it does not test whether it is meaningful. In the same ways as the sentence 'Colourless green ideas sleep furiously' is grammatically correct but meaningless, so too can dBase expressions be.

TEST() in dBase II and TYPE() in dBase III work a little differently. TEST determines whether an expression is syntactically valid, and if so, what type of data it contains. If the expression is valid and numeric, it will return -6, if it is valid and logical it returns 1, and if it is valid and a character function, it returns its length. However, if the expression is non-parsable, TEST will return 0.

TYPE determines whether a character expression (the argument of the expression must be placed in quotation marks) is valid, and if so, what type of data it contains. For valid expressions it returns an N if numeric, a C if character, an L if logical and an M if memo. If the expression is invalid or the variable used in the expression does not exist, U (for undefined) is returned.

The parsability of an expression has nothing to do with its value. dBase does not have a function for testing the actual validity of a command. So, for example, the command line:

```
? 0 <> TEST ( a = b )
```

dBriefing

Feedback and Food for Thought

Dear dLetter,

First let me thank you for putting out such an excellent newsletter that fills a need in the dBase community. I wish you every success with it.

I have discovered another of dBase's 'undocumented features' which can produce totally baffling results. I am using dBase version 2.4 on a 128 Kbyte Microbee, and the problem happens when a command file contains a line longer than about 220 characters and is called by another command file. When dBase encounters a RETURN statement in the second command file it forgets to return to the calling program and instead it returns to the dot prompt. As you can see, the result is unrelated to the cause.

This bug is very difficult to get around; for example:

```
SUM amount TO p:sum FOR ;
 category=p:count .AND. ;
 VAL($(date,7,2)+$(date,4,2)+;
 $(date,1,2) > VAL($(p:from,7,2)+;
 $(p:from,4,2)+$(p:from,1,2))-1;
 .AND. VAL($(date,7,2)+$(date,4,2);
 +$(date,1,2) < VAL($(p:to,7,2)+;
 $(p:to,4,2)+$(p:to,1,2))+1
```

This command line totals the value in the amount field of all records in a certain category and between two dates. With a large database this is a slow process but to rewrite it using a DO WHILE loop would make it extremely slow.

So, if your program stops at the end of a subroutine, look for a line longer than 220 characters and be careful that a macro substitution doesn't produce an offending line (that would be a fun one to find). Like a lot of dBase bugs, this one will sometimes work properly.

I use the FILE() function extensively in my programming and it hasn't bitten me yet (as was described in the August '85 issue of dLetter). This may be because the bug has been fixed in version 2.4 or the way the 128 Kbyte Microbee stores the disk directory in memory. Maybe I'm being foolhardy.

Have you any plans to set up a dBase public access system where dBase users could upload and download their programs? I have a bulletin board program written in dBase by Gary Woodman and myself which would suit such a system (Gary lives in Darwin and this program is another success of the public access system).

Dean Cording
Lismore, NSW

We're hoping to place as many dBase programs as possible on the Your Computer Bulletin Board (phone (02) 662 1686). There are already a number of dBase programs up there, but you need to be a member of the board to get access. The Your Computer board's software is also all written in dBase (as you'll notice sometimes when the system develops a bug, and you get a message such as 'SYNTAX ERROR').

The Sydney dBase User's Group has a library of public domain programs which are available to group members, either on disk, or on a special section of the PC User's Group bulletin board. See the user's group section of this newsletter for the number.

Dear dLetter,

I noticed the suggestion by L.F. Roberts in issue 6 of dLetter regarding setting the dBase date from the system date in a NEC APC. I have been using a similar idea on my CP/M Plus system for some time and I include a listing of a dBase command file, SHOWDATE.CMD, to retrieve the system date and time and reformat it into character strings. Note that the day of the week can be obtained simply and is useful, along with the date and time for report headings. The listing is commented so those parts not needed in an application may be left out.

Incidentally, I have rewritten the CP/M Plus DATE utility to

```
*SHOWDATE.CMD   -   13/11/85
*Author: Grahame Davis
*Purpose: dBase command file to retrieve CP/M-3 (CP/M Plus) system
*         date and time and reformat into strings.

SET TALK OFF
STORE 41984 TO pataddrs
SET CALL TO pataddrs

*Routine to read system date and time
POKE pataddrs,126,254,5,216,229,35,235,14,105,205,5,0
POKE pataddrs+12,225,17,5,0,25,119,201

*Read sytem date and time into 5-character string.
*First two characters are 16-bit days since 1-JAN-78,
*remaining three characters are hours, minutes and seconds each
*stored as tow BCD digits
STORE '     ' TO string
CALL string

*Extract days since 1-JAN-78
STORE RANK($(string,2,1))*256+RANK($(string,1,1))-1 TO days

*Calculate day of week
STORE days-INT(days/7)*7 TO tempn
STORE TRIM($('Sun   Mon   Tues  WednesThurs Fri   Satur ',;
tempn*6+1,6)+'day' TO weekday

*Calculate number of leap year cycles
STORE INT(days/365*4+1) TO tempn
STORE tempn*4+78 TO years
STORE days-(tempn*(365*4+1)) TO days

*Calculate years in current leap year cycle
STORE 1 TO tempn
STORE '365365366365' TO temp
DO WHILE days >= VAL($(temp,tempn,3))
   STORE days-VAL($(temp,tempn,3)) TO days
   STORE years+1 TO years
   STORE tempn+3 TO tempn
ENDDO

*Calculate month in current year
STORE 1 TO months
STORE 1 TO tempn
STORE '312831303130313130313031' TO temp
STORE VAL($(temp,tempn,2)) TO tempnl
DO WHILE days >= tempnl
   STORE days-tempnl TO days
   STORE months+1 TO months
   STORE tempn+2 TO tempn
   STORE VAL($(temp,tempn,2)) TO tempnl
   IF (months = 2) .AND. (years = INT(years/4)*4)
```

accept the date in DD/MM/YY or DD-MM-YY format and patched the DIR utility to display dates in DD/MM/YY format. Since there is no longer a CP/M column in the magazine, I had not bothered to send in details. However, if anyone is interested I would be happy to send in a DATE.COM and patch details for DIR.COM on disk.

G.E. Davis
Carlingford, NSW

Dear dLetter,

I would like to thank you for the hints and tips which appear in your newsletter. As I have used a couple of them in my programs, I would like to reciprocate by offering a small program which might be beneficial to some users.

dBase II runs faster if the index files are on a different drive from the .dbf files - and in dBase any increase helps!

The program, MDRIVE.PRG, can be very useful for anyone using a hard disk or going to upgrade to one who decides to change the location of the data and index files from one drive to another - or even better to a memory drive (RAMdisk). The program is run before the main menu for any application.

If the memory file MDRIVE.MEM does not exist on the current drive, the program will prompt the user to enter the drive names, and will save them on the

Management Technology Education Pty. Ltd.

MTE is Australia's leading supplier of quality PC training to business and government.

MTE courses feature top-line instructors, "hands-on" format, professional design, premium quality training facilities, Melbourne and Sydney City Centre locations.

When you have the training need, we have the course.

- PC's as a Business Tool
- Multiplan Introduction & Advanced
- dBASE II & III Intro., Intermediate & Advanced
- WordStar, Introduction & Getting More From WordStar?
- LOTUS 1-2-3 & SYMPHONY

For full details of the above and other MTE PC Courses phone for our **FREE** comprehensive booklet or complete and send the coupon opposite.

No. 1 PC TRAINER

SYDNEY
2nd Flr, 31 Market Street,
Tel: (02) 290 3555.

MELBOURNE
6th Flr, 99 Queen Street
Tel: (03) 67 7117.

```
      STORE tempn1+1 TO tempn1
   ENDIF
ENDDO

*Calculate time of day
STORE RANK($(string,3,1)) TO hours
STORE INT(hours/16) TO tempn
STORE hours-(tempn*16)+(tempn*10) TO hours
STORE RANK($(string,4,1) TO mins
STORE INT(mins/16) TO tempn
STORE mins-(tempn*16)+(tempn*10) TO mins
STORE RANK($(string,5,1) TO secs
STORE INT(secs/16) TO tempn
STORE secs-(tempn*16)+(tempn*10) TO secs

*Prepare for displaying date and time
STORE $(STR(days+101,3),2,2) TO day
STORE $('JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC',months*3-2,3) TO month
STORE $(STR(years+100,3),2,2) TO year
STORE $(STR(hours+100,3),2,2)+':'+$(STR(mins+100,3),2,2);
      +':'+$(STR(secs+100,3),2,2) TO time

*dBase date format (may be used to set date).
STORE day+'/'+$(STR(months+100,3),2,2)+'/'+year TO date

*Display it all
? "Today is &weekday, &day-&month-&year or &date and the time is &time
RETURN
```

disk. Now any dBase program can be written without worrying on which drive to put the dbf or ndx files.

For example, if the file NAMES.DBF contains names of customers and should be indexed on customers' postcode to NAMPOST.NDX, the following process will do it:

```
DO MDRIVE
STORE 'NAMES' TO mnames
STORE 'NAMPOST' TO mnampost
SELECT PRIMARY
USE &drive1&mnames
INDEX ON postcode TO
    &drive2&mnampost
RELEASE ALL LIKE mnam*
```

If the system has only two drives, the answers will be A: and B: to the MDRIVE.PRG questions, and these will be saved to disk. These questions will be asked only once when initialising — if the system is upgraded to hard disk, all that has to be done is to delete the file MDRIVE.MEM. On the first run, the program will force allocation of new drives and there won't be any need to edit any .PRG files.

Eli Light
South Yarra, VIC

Thanks for the program. One word of warning: anyone who is using a version of dBase II prior to 2.41 should think twice about using the FILE() function. In early releases of dBase this function was bug ridden in a big way: using it while you had a database

No. 1 dBASE TRAINER

SYDNEY
2nd Flr, 31 Market Street,
Tel: (02) 290 3555.

MELBOURNE
6th Flr, 99 Queen Street
Tel: (03) 67 7117.

Management Technology Education Pty.Ltd.

When you need dBASE training call MTE, Australia's largest supplier of quality PC training.

dBASE courses include:
- dBASE II and dBASE III Introduction
- dBASE II and dBASE III Intermediate
- dBASE Advanced with Les Bell

SEND NOW FOR YOUR FREE COMPREHENSIVE COURSE BOOKLET

☐ Please send me FREE the MTE comprehensive PC course booklet.

Name: ....

Position: ....

Company: ....

Address: ....

.... Postcode: .... Phone: ....

file open could cause your disk directory to be completely rewritten. Although it has been fixed, we're still chary of using it, as it munched 32 files on our system early on, and we haven't quite recovered our faith.

Dear dLetter,

Here is another date program, FINDDATE.PRG, that may be of use. It accepts a date string (8 characters) and desired interval (in days - plus or minus), and returns the new date. The program could be put in two halves, such that the first half would allow days between dates to be found, using the value 'D' as the number of days since 1/01/1900.

Also, I am using dBase II under CP/M-86 and it appears 2.43* will not be available; any news of Ashton-Tate dropping support for this version?

Norman Wheeler
Greenacre, NSW

Evidently 2.43* will not be made available for CP/M-86. Versions for 8-bit CP/M and PC-DOS are already on the market here.

dGenerate Report Spooling for dBase III Plus

By Tom Retting

There are situations where it is inconvenient to print reports in real time. For example, a company needs to print an order acknowledgment form and one or more labels for each order entered. They could either use two printers, requiring an additional capital investment, or spool the labels and the acknowledgments for printing later.

Another example is a corporation with a large mainframe COBOL application converted to dBase III on an IBM PC/AT. The reports are so voluminous that printing them requires a printer capable of tremendously high rates of speed. The corporation already has several high-speed printers in another department, so this application spools the reports for printing by the other department.

This article presents a procedure that can be called with a syntax similar to what we would use with the @...SAY command. The formatted output command in dBase, @...SAY, is not designed to send output to a disk or ALTERNATE file. In dBase II, for instance, you could SET FORMAT TO PRINT and force the output of @...SAYs into a file.

With versions 1.0 and 1.1 of dBase III, we can use the unformatted output command - the question mark - to create a disk file, as long as we do not need to format the data during output (Only the @...SAY command gives us the PICTURE/FUNCTION formatting options in versions 1.0 and 1.1 of dBase III.) With the dBase III Plus, we can use the TRANSFORM() function, which brings all the power of PICTURE/FUNCTION to the unformatted output mode.
The procedure presented below is designed for dBase III Plus but will work with any version of dBase III. To set up, first, place the following Atsay PROCEDURE into your procedure file. Next, in the calling program SET ALTERNATE TO <filename> and initialize the following memory variables.

```
   max_row : numeric, maximum
      number of rows per page
   max_col : numeric, maximum
     number of columns per row
   prev_row : = 0
   prev_col : = 0
```

Then, replace each @...SAY and @...GET that you want to capture to the alternate file in your report program with an "DO Atsay WITH..." statement. For example,

```
 @ 5,6 SAY Name PICTURE "!!!!!"
```

becomes:

```
DO Atsay WITH 5, 6, Name,"!!!!!"
```

and

```
 @ 5,6 SAY "Name" GET Name
```

dBASE II Users are laughing
Now they have dTOOLKIT
Containing many dBASE III facilities & more

- **dLIBRARY - OBJECT CODE ROUTINES:** Arrays, extra 256 Memory Variables, dBASE III Data Facilities, Amount in Words, DOS Type, Soundex Codes, Check Digits, Random Numbers, SQRT, Tolower, Capitalize, Isalpha, Encrypton, compression...and Many More. List $55.00
- **dBASIC - FUNCTIONS:** Process dBASE II Data and Index Files using CB80 or CB86 Compiled BASIC. List $45.00
- **dFIX - UTILITY:** Fix Corrupted Data Files List $40.00
- **dCRYPT - UTILITY:** Encrypt and Decrypt Data Files List $30.00
- **dMENU -** - dBASE RUNTIME MENU SYSTEM: Maintenance Routines, Menu Hierarchy, Security. List $40.00

*CP/M-80, CP/M-86, MS-DOS, PC-DOS
*depending on disk format

TOTAL PACKAGE $99

Available from:
The Computer Trader,
98 Hoddle Street,
ABBOTSFORD, 3064
Telephone Orders:
(03) 417 7044

```
* PROCEDURE  Atsay
* Author...: Tom Rettig
* Date ....: November 1, 1985
* Version..: dBASE III Plus
* Note(s)..: Procedure for sending formatted output to an
*            ALTERNATE file.
*
PARAMETERS at_row, at_col, exp, pic
*
* ---Trap "out of range" errors.
IF at_row > max_row .OR. at_col > max_col
   * ---Message goes to screen only.
   SET ALTERNATE OFF
   IF at_row > max_row
      ? "Row is out of range -->", at_row
   ENDIF
   IF at_col > max_col
      ? "Column is out of range -->", at_col
   ENDIF
   SET ALTERNATE ON
   RETURN
ENDIF
*
SET CONSOLE OFF
*
* ---Branch if current row is less than previous row.
IF at_row < prev_row
   *
   * ---Do a formfeed in the file.
   at_count = 0
   DO WHILE at_count < ( max_row - prev_row )
      ?
      at_count = at_count+1
   ENDDO
   *
   prev_row = 0
ENDIF
*
* ---Skip to current row.
at_count = 0
DO WHILE at_count < ( at_row - prev_row )
   ?
   at_count = at_count+1
ENDDO
*
* ---Branch if current column is less than previous column.
IF at_col < prev_col
   * ---Do a linefeed in the file.
   ?
ENDIF
*
* ---Skip to current column.
?? SPACE( at_col )
*
* ---Output the expression in the format of the PICTURE.
?? TRANSFORM( exp, pic )
* ---Version 1.0 and 1.1 users remove the TRANSFORM() function.
*
* ---Save current row and column.
prev_row = at_row
prev_col = at_col + LEN( TRANSFORM( exp, pic ) )
* ---Version 1.0 and 1.1 users remove the TRANSFORM() function.
*
SET CONSOLE ON
RETURN
*
* EOP Atsay.PRG
```

becomes:

```
 DO Atsay WITH 5,6,"Name",""    or
 DO Atsay WITH prev_row, ;
    prev_col+1,Name,""
```

Your calling program should look something like the following:

```
* ---In the report procedure.
SET ALTERNATE TO <filename>
SET ALTERNATE ON
STORE 0 TO prev_row, prev_col
max_row = 23
max_col = 79
* ---Report algorithm goes here.
* ---Syntax used in place of
* ---the @...SAY command:
* ---DO Atsay WITH <row>, <column>
* ---         <expression>, <picture>
*
CLOSE ALTERNATE
RETURN
* End of report procedure.
```

The text file containing the spooled report can be printed by using the PC/MS-DOS PRINT.COM, COPY <filename> PRN, TYPE <filename> > LPT1, or from other word processing software.

Copyright 1985 Ashton-Tate. Reprinted from Technotes, with permission from master Australian distributor, Arcom Pacific.

dFixer

Shifted Data Displays

by Oliver Biggerstaff

A database file may become corrupted for any number of reasons. Often the corruption may be the form of shifted data in full-screen edit screen displays. This is caused by an embedded null character in a record. A null character is represented as a 00 hex and is used by dBase II and dBase III as a string terminator. A string terminator is a character that can be thought of as a delimiter, much like double quotes surrounding a character string, or as a carriage return and linefeed at the end of a record.

The dBase APPEND, EDIT, BROWSE, and other full-screen commands work on the principle that the cursor's positioning on the screen depends on certain attributes, such as the length of a field and its current position

The appearence of shifted data is caused by the embedded string terminator forcing dBase to stop the display of a field prematurely, placing the cursor at an incorrect location. If a null character is encountered before all the characters of a field have been displayed, dBase will stop listing that particular field and will produce the shift effect by displaying the next field at an incorrect screen location.

The data is actually not shifted physically in the database file. It is simple, therefore, to correct the database by replacing any null character with another character that does not force dBase to display incorrectly. A good choice for this character is the ASCII character zero (0) or 30 hex. Replacing a null with this character is advantageous for two reasons:

1. Since dBase can display it, you can locate the corrupted data.

2. A zero character will have the least disruptive effect on the contents of the database file. Additionally, logical fields that contain a zero character will be displayed as (.F.)

Once all the null characters have been replaced, the BROWSE or EDIT commands can be used to retype the original data in place of the characters that replaced the nulls.

The following is one of many methods that can be used to replace null characters with other characters. In this example we use the PC/MS-DOS utility DEBUG.COM, found this program on the supplemental PC/MS-DOS disk. These examples assume that the database file is 64K or less in size. Refer to the PC/MS-DOS manual for more information on DEBUG and how to use data segments if the database file is larger than 64K.

The contents contained in the < and > symbols must be calculated by you, and entered without the symbols. For example, if the value of the CX register is 2C80H, then <CX+100H> is to be replaced with 2D80H. Be sure that before you attempt this procedure, you have made a backup of your database file.

Unfortunately, this method of replacing null characters can be very tedious if many characters must be replaced. For those of you with a large amount of corrupted data, it is suggested that you use other well-known utility programs such as NIBBLER, NORTON UTILITIES, JAZ, or PATCH. These programs will allow you to look at very large database files directly from the hard disk. Some of these programs may also have a global search and replace option.

For dBASE II users on a 16-bit computer:

```
C>DEBUG <database>.DBF ;Read database file into memory.

-RCX                   ;Get the value in the CX
                       ;register.
-S 309 <CX+100H> 00    ;Search for nulls in the file.
   .
   .                   ;A list appears here of one or
   .                   ;more addresses containing a
   .                   ;null.
-E <address> 30        ;Replace each individual address
                       ;that contains a null with a
                       ;zero.
-W                     ;Save the modifications to disk.
-Q                     ;Quit DEBUG.
```

For dBASE III users:

```
C>DEBUG <database>.DBF ;Read database file into memory.

-RCX                   ;Get the value of the CX
                       ;register.
-S 100H 1121H 0DH      ;Search for the end of the
                       ;header for address containing
                       ;0DH
xxxx:yyyy              ;for address containing 0D.
                       ;Search for null characters.
-S <yyyy+2> <CX+100H> 00
   .
   .                    ;List of addresses containing
   .                    ;nulls.
   .
-E <address> 30H        ;Replace each null with a zero.
-W                      ;Save the modifications to
                        ;disk.
-Q                      ;Quit DEBUG.
```